IN THE CLAIMS:

The text of all pending claims, (including withdrawn claims) is set forth below. Cancelled and not entered claims are indicated with claim number and status only. The claims as listed below show added text with underlining and deleted text with ~~strikethrough~~. The status of each claim is indicated with one of (original), (currently amended), (cancelled), (withdrawn), (new), (previously presented), or (not entered).

Please AMEND claims 1-3 and 6-7 in accordance with the following:

1. (CURRENTLY AMENDED) A method, using a host CPU, for co-verifying hardware and software~~, by using a host CPU,~~ for a semiconductor device on which at least one target CPU and one OS are mounted, the hardware/software co-verification method comprising the steps of:

(a) inputting, as a verification model, a timed software component described in a C-based language and compiling the same, inputting, as a verification model, a hardware component described in the C-based language and compiling the same, and linking together the compiled timed software component and the compiled hardware component, wherein an interrupt routine scheduler is input that is equivalent to an interrupt processing section of an instruction set simulator but is provided as an independent unit;

(b) inputting a testbench and compiling the same;

(c) linking together the verification models ~~processed~~ input in step (a) and the testbench ~~processed~~ input in step (b);

(d) performing a C-based native code simulation without per-instruction interpretation and execution, based on an executing program generated in step (c); and

(e) outputting a result of the simulation performed in step (d).

2. (CURRENTLY AMENDED) A method, using a host CPU, for co-verifying hardware and software~~, by using a host CPU,~~ for a semiconductor device on which at least one target CPU and one OS are mounted, the hardware/software co-verification method comprising the steps of:

(a) inputting, as a verification model, a timed software component constructed from binary code native to the host CPU, inputting as a verification model, a hardware component described in a C-based language and compiling the same, and linking together the input timed software component and the compiled hardware component, wherein an interrupt routine scheduler is input that is equivalent to an interrupt processing section of an instruction set

simulator but is provided as an independent unit;

(b) inputting a testbench and compiling the same;

(c) linking together the verification models ~~processed~~ input in step (a) and the testbench ~~processed~~ input in step (b);

(d) performing a C-based native code simulation without per-instruction interpretation and execution, based on an executing program generated in step (c); and

(e) outputting a result of the simulation performed in step (d).

3. (CURRENTLY AMENDED) A method, using a host CPU, for co-verifying hardware and software~~, by using a host CPU,~~ for a semiconductor device on which at least one target CPU and one OS are mounted, the hardware/software co-verification method comprising the steps of:

(a) inputting, as a verification model, a timed software component described in a C-based language and compiling the same, inputting, as a verification model, a timed software component constructed from binary code native to the host CPU and compiling the same, inputting, as a verification model, a hardware component described in the C-based language and compiling the same, and linking together the compiled or input timed software components and the compiled hardware component, wherein an interrupt routine scheduler is input that is equivalent to an interrupt processing section of an instruction set simulator but is provided as an independent unit;

(b) inputting a testbench and compiling the same;

(c) linking together the verification models ~~processed~~ input in step (a) and the testbench ~~processed~~ input in step (b);

(d) performing a C-based native code simulation without pre-construction interpretation and execution, based on an executing program generated in step (c); and

(e) outputting a result of the simulation performed in step (d).

4. (PREVIOUSLY PRESENTED) The hardware/software co-verification method as claimed in claim 1 wherein, in order to generate in advance the timed software component described in the C-based language from an untimed software component described in ANSI-C, the method further comprises the steps of:

inputting the untimed software component described in ANSI-C, and recognizing basic blocks and inserting control points;

generating binary code native to a target CPU by compiling the untimed software

component in which the control points have been inserted;

computing execution time between the control points in the generated binary code native to the target CPU; and

inserting, in accordance with the computed execution time, an execution time insertion statement at each of the control points inserted in the untimed software component, and thus outputting the timed software component described in the C-based language.

5. (PREVIOUSLY PRESENTED) The hardware/software co-verification method as claimed in claim 2 wherein, in order to generate in advance the timed software component constructed from the binary code native to the host CPU from an untimed software component constructed from binary code native to a target CPU, the method further comprises the steps of:

inputting the untimed software component constructed from the binary code native to the target CPU, and converting the same into a software component expressed in the binary code native to the host CPU;

recognizing basic blocks and inserting control points in the software component expressed in the binary code native to the host CPU;

computing execution time between the control points in the software component in which the control points have been inserted; and

inserting, in accordance with the computed execution time, binary code functionally equivalent to an execution time insertion statement at each of the control points inserted in the software component, and thus outputting the timed software component constructed from the binary code native to the host CPU.

6. (CURRENTLY AMENDED) The hardware/software co-verification method as claimed in claim ~~1~~ 3 wherein, in order to generate in advance the timed software component described in the C-based language from an untimed software component described in ANSI-C, the method further comprises the steps of:

inputting the untimed software component described in ANSI-C, and recognizing basic blocks and inserting control points;

generating binary code native to a target CPU by compiling the untimed software component in which the control points have been inserted;

computing execution time between the control points in the generated binary code native to the target CPU; and

inserting, in accordance with the computed execution time, an execution time insertion

statement at each of the control points inserted in the untimed software component, and thus outputting the timed software component described in the C-based language.

7. (CURRENTLY AMENDED) The hardware/software co-verification method as claimed in claim ~~2~~ 3 wherein, in order to generate in advance the timed software component constructed from the binary code native to the host CPU from an untimed software component constructed from binary code native to a target CPU, the method further comprises the steps of:

inputting the untimed software component constructed from the binary code native to the target CPU, and converting the same into a software component expressed in the binary code native to the host CPU;

recognizing basic blocks and inserting control points in the software component expressed in the binary code native to the host CPU;

computing execution time between the control points in the software component in which the control points have been inserted; and

inserting, in accordance with the computed execution time, binary code functionally equivalent to an execution time insertion statement at each of the control points inserted in the software component, and thus outputting the timed software component constructed from the binary code native to the host CPU.